Computer Science

WHY SCALABLE DISTRIBUTED MULTI-AGENT SIMULATION IS HARD

Quincy E. Mitchell, Les Gasser*, Kiran Lakkaraju, Computer Science Department and Graduate School Library and Information Science, University of Illinois, at Champaign-Urbana, IL 61801, gasser@illinois.edu

An agent is an abstract structure that has actions, the ability to learn, and a common environment that is shared with other structures including other agents, and is where they interact. Many research topics can be explored with agent-based modeling; we are using it for simulating language evolution, as well as information processes in biological systems. Our models and experiments require scalable high level languages and tools, but these do not yet exist. Specifically, our research requires distributed, scalable simulations that model complex messages transmitted among social groups of agents that are networked in structured and changing ways. At runtime, agents will need to decide where to move, what to do, how to do it, with whom to interact, how to “talk”, and what to say, all in a simulated world environment that remains consistent for all agents. There are three core conceptual problems for tools supporting these activities: concurrently running agents in complicated structures, maintaining consistency across a distributed world, and maintaining consistency of time and events such as interactions and messages.

We have been learning about, studying, experimenting with, and evaluating several possible tools to underpin these simulation frameworks, including Unified Parallel C (UPC), concurrent RePast exploiting Terracotta clustering middleware, and the possibility of using the Blue Gene/Blue Waters (petascale) hardware architectures.

UPC is a Single Program Multiple Data (SPMD) parallel extension to the C language. Under SPMD models, each thread runs the same instructions in the same program, and data itself causes different results to be created in different threads. UPC contains many features to split instructions among threads and synchronize them. UPC's power is in a virtual shared space it offers for all threads. Programmers need not be concerned with how threads are allocated to CPUs. The major issue we’ve found with UPC is that lack of support for sharing complex data structures while maintaining effective distribution. Though UPC is powerful, is it not a very high level language and complex data structures (like agents or worlds) are difficult to control.

Terracotta (TC), was specifically created to share complex data structures. However, TC was also specifically developed using a simple master-worker paradigm. TC maintains its shared software space only on one computer, that functions as the “brain” of the master. When this master directs its slaves, it splits a task into independent components runs them concurrently, and re-assembles the partial results, with no assumption of interactions among the sub-parts at runtime. This interaction-free, master-worker structure creates a huge hurdle that makes it difficult to achieve our aims of scalable, agent-interactive simulations. We need agents to directly interact with one another via worlds and messages.

Since at present no framework fully addresses our needs, the remainder of our presentation discusses possible new framework features and approaches to some of the basic issues outlined above. We discuss the general problems with creating a new framework that handles direct interactions among agents running in parallel in shared worlds, on scalable hardware platforms such as clusters (ala Terracotta) and special purpose machines (ala Blue Gene/Blue Waters).

Quincy Mitchell was supported by NSF-REU Award No. IIS-0429082